Towards Usable Off-Chain Payments

Guillaume Felley

Supervised by:
Dr. Arthur Gervais
Prof. Dr. Roger Wattenhofer

September 2018

.

Acknowledgement

I would first like to thank my thesis supervisor Dr. Arthur Gervais of the Department of Computing at Imperial College London. He followed my work from the start until the end. He was always willing to take the time to understand my progress and advise me for further directions.
I will give special thanks to my friends Thibault Meunier and Rami Khalil with who I collaborated closely during this project.
Additionally, I would like to thanks Humble Team, a design agency that provided me with the graphical assets for the mobile app and Fedir Ushakov for his help and expertise on mobile application front-end development.
Finally, I would like to address special mentions to my family for their constant support and Carmen for her encouragements and help proof-reading my grammar mistakes.

Guillaume Felley
London, 17.09.2018

Contents

1 Introduction

Blockchains such as Bitcoin or Ethereum are distributed append only ledgers that rely on a full consensus over each of the updates of the ledger. These updates are called blocks and are hash chained together by including in each block a hash of the previous block. It is impossible to alter a block without altering all the following blocks. The major innovation that made the success of Bitcoin was the Nakamoto consensus[44] that uses proof of work to allow an unknown number of mistrusting peers to reach consensus. This mechanism has proven to be surprisingly resilient against all sort of attacks. However, this level of security comes at a very high cost, since each node in the network is required to verify the validity of the complete ledger starting from the genesis block up to the current date. Every new block periodically produced has to be broadcasted to all nodes in the network. Blockchains such as Bitcoin have shown their limits in terms of scalability. In December 2017 transaction fees for a Bitcoin transaction reached a peak of 55 USD, while the current fee in August 2018 is around 0.50 USD[1]. Surge in transaction fees is being observed when the demand is higher than the maximal throughput that the ledger can provide. Bitcoin and Ethereum offer a maximum of 7 and 20 transactions per second, respectively. It was shown that proof of work blockchains can hardy scale beyond 60 transactions per second without weakening considerably their security[35]. These limitations make applications such as micro-transactions or mainstream adoption of cryptocurrencies difficult.

In this context, some people have proposed to change the paradigm: instead of broadcasting every transactions in the Blockchain, transactions are made off-chain and the ledger is used only in case of discourse to resolve conflicts. The Blockchain would be used only occasionally when the peers disagreeing on the outcome of a transaction. The first challenge in this approach is to provide an efficient conflict resolution mechanism to guarantee that no party can cheat. These approaches are built on the top of the existing Blockchain leveraging the Bitcoin scripting language or the Ethereum virtual machine and are therefore called second layer protocol. In this work we will look at two different ways of building these second layer protocols. The first protocol is a payment channels network, with its most famous implementation, the Lightning Network for Bitcoin. The second type of protocols are payment channel hubs such as NOCUST[39] that leverage the power of Turing Complete smart contract languages on platforms like Ethereum.

However, these potential scaling solutions come with a new set of constrains that highly impact user experience. The current state of the art is only accessible to advanced users, the Lightning Network is only fully usable through a command line interface and requires its users to have an always online server. These solutions get their security properties from actively monitoring other parties behaviour and to be willing to take actions if a fraud is detected. Additionally, off-chain payments need to be collateralised and an inadequate or unbalanced

[1] https://bitinfocharts.com/comparison/bitcoin-transactionfees.html

amount of collateral can lead to payment failure. These constraints are new and do not exist when doing regular Blockchain transactions, therefore it brings new challenges to make the technology usable. In this work there is a particular focus on wallet software. Wallets hold the funds of users by managing private keys. This is the critical user facing part of a Blockchain infrastructure. Security is the top priority for wallets, although they still need to be simple enough to be accessible for non-technical users. Support of off-chain scaling solution in wallet software is a complex task that deteriorates the user experience. Tasks such as channel management or the requirement to maintain off-chain balances add complexity and raise the entry barrier for cryptocurrency users. Security monitoring has to be handled with care by the wallet as it can lead to losses of funds. Additionally, payment failure due to routing issues or liquidity issues are currently frequent.
On the first place, some background explaining the underlying mechanisms of payment channel network and payment hubs like NOCUST[39] is presented. In section 2 detailed informations on the current state of the Lightning Network is given, together with data on the adoption and on the current network Topology. Existing Lightning Network wallet were investigated, evaluating their capabilities and their security. In section 3 we look at the current implementation of a NOCUST hub deployed on the Ethereum main network. Additionally, some results on performance and gas efficiency of the NOCUST hub are provided. In the last chapter we will present the Mobile Wallet developed for the Liquidity Network supporting off-chain payments through a NOCUST payment hub.
The contributions of this work by order of importance are: the first fully bi-directional mobile wallet supporting off-chain payments for the Ethereum main network, an analysis of the current state of the Lightning Network topology, a survey of the capabilities of existing wallets for the Lightning network, an analysis of NOCUST's operational gas costs and a user study of wallet expectations in terms of user experience.

2 Background on Off-chain payment solutions

2.1 Payment Channel network

The idea of payment channels is not recent, it was first conceptualised by Satoshi Nakamoto, creator of Bitcoin in its early days[24]. The approach consist of locking up an amount of collateral on the Blockchain to back future off-chain transactions.

2.1.1 Payment channel

A payment channel consist of locking up funds of two parties in a special address. It will be required to use the blockchain and to pay the network fee only when opening and closing the channel. Between the opening and the closing of a channel, the parties can make a potentially unlimited number of off-chain

transactions.
To build a payment channel several building blocks such as Multi-signature, Time-Lock contract and Hash locked contract are needed.

Multi-signature : It is a special address that requires several private keys to unlock its funds. For a payment channel we will use a 2-of-2 multi-signature, that require the approval of the 2 parties to spend the funds in the address.

Time-lock contract : Makes the funds of this address spendable only at a given time in the future, at certain block height.

Hash-lock contract Makes the funds of this address spendable only if the preimage of a specific hash is provided.

By combing Time-lock contract and Hash-lock contract a Hash Time Locked Contract or HTLC can be created. The funds in this contract will either be sent to a specific address after a given time or to a different address if the correct preimage of a given hash is provided. Payment channel networks are based on exchanging HTLCs off-chain between the 2 parties involved in a channel to update the off-chain state. The exchanged HTLCs are constructed in a way to incentive the parties to collaborate. If a party tries to force the closure of a channel without the other party's collaboration he will see his funds locked for a while before getting them back. If a party tries to cheat, he takes the risk of losing all his funds.
These are the steps composing the life cycle of a payment channel:

Channel opening : To setup a bidirectional payment channel, both parties involved must first agree on an opening transaction. The opening transaction determines how much funds will be sent in this channel. To open the channel Alice and Bob send 5 Bitcoins each one to a 2-of-2 multi-signature address. This is the opening transaction.

Additionally, Alice and Bob both generate a secret and exchange its hash.

Alice now creates a subsequent transaction from the opening transaction called the "commitment transaction". With the commitment transaction, Alice sends 4 Bitcoins to herself and 6 Bitcoin to a HTLC. It can be unlocked by Bob on his own, but only after a certain amount of blocks, for instance 1000 blocks (Time-locked). Alternatively, it can be unlocked by Alice if she knows the secret value for which the hash is equal to the value that Bob just gave her (Hash-locked). Of course, Alice cannot make use of the second option as only Bob is in possession of the secret value. Alice signs her half of the commitment transaction and gives it to Bob.

Meanwhile, Bob does the same action in a mirrored way. He creates a commitment transaction from which he sends 6 Bitcoins to himself, and four to a HTLC. Alice can unlock this address if she waits 1000 blocks or Bob can unlock with Alice secret. Bob signs his half of the transaction and gives it to Alice.

After this exchange of half-valid commitment transactions and the hashes of the secrets they both sign and broadcast the opening transaction on-chain funding the multi-signature.

At this point both parties could sign and broadcast the half-valid commitment transaction that they got from the other. If Bob does, Alice gets 4 Bitcoins immediately and Bob gets 6 Bitcoins after 1000 blocks. If Alice does, Bob gets 6 Bitcoins immediately and Alice gets 4 Bitcoins after 1000 blocks. Whomever signs and broadcasts the transaction will have to wait 1000 blocks. This mechanism ensures that both parties can always recover their funds even if one party never comes back online. The reason why the broadcasting party needs to wait 1000 blocks will be explained further.

Updating the Channel : After a while Bob and Alice wish to make an off-chain transaction. Bob wants to send back one Bitcoin to Alice. First, both parties need to repeat the process described previously to create a new HTLC with updated values. This time, both Alice and Bob attribute themselves 5 Bitcoins, and both attribute 5 Bitcoin the HTLC. The conditions to unlock the HTLC are similar except that they require a new secret and the balances are changed according to the new Bitcoin distribution. They both sign this new HTLC and give it to each other.

Now, Alice and Bob exchange the secret from the First HTLC.

At this point, again, both parties could sign and broadcast the new HTLC they just got. Their counter party would get 5 Bitcoins immediately, while the broadcaster would have to wait 1000 blocks. Therefore, the channel is updated.

It is now important to understand what is blocking Bob from broadcasting the old HTLC in which he received 6 Bitcoins. Alice is now in possession of the secret allowing her to get these 6 Bitcoin for herself. Because Bob needs to wait 1000 blocks before getting the 6 Bitcoins, Alice has plenty of time to broadcast the "poisoning" transaction using Bob's secret to get the 6 Bitcoins for herself getting all of Bob's funds. Therefore, Bob is strongly disincentivised to broadcast an old state at his advantage.

It is important to note that for security, the parties need to constantly watch the Blockchain. If a party goes offline for an extended period of time, more than 1000 blocks, the second party can attempt to submit an outdated state to steal funds. Both parties need to be online and willing to submit the poisoning transaction if a fraud is detected.

2.1.2 Linked payments

We could imagine a situation where Bob has an open channel with Alice and another one with Charlie. If Alice wants to send a payment to Charlie she can open a channel with Charlie as they do not have a channel together. The drawback is that she would have to pay the fees for opening a channel, moreover

she might need to transact only once with Charlie. In this case using a payment channel is not interesting. Another possibility would be to send the payment through the channel of Bob, as he is connecting both. Alice would send a payment to Bob and Bob would forward it to Charlie. In this naive scenario we would need to trust Bob for forwarding the payment and not keeping the money for himself. However, using HTCLs and passing a hash preimage from the recipient to the sender we can execute the linked payment in a trust free manner. The payments across the links are enforced atomically, either all payments are successful or all fail. Therefore, nobody can loose their funds. Under certain conditions, the worst that can happen is to pay the network fees and to see funds lockup for a period of time. If all the parties collaborate, the payments happens only off-chain and do not require to pay the blockchain network fee. This linked payments can have several hops, if Alice and Charlie do not have an open channel with a person in common such as Bob, we could imagine a longer route for the payment: `Alice -> Mike -> Dave -> Charlie`
The idea behind projects such as Lightning is to create a network of payment channels to facilitate the creation of linked payment between entities that do not necessarily know and trust each other. This needs a protocol to open channels, advertise them and specify how to exchange HTLC to allow linked payments. This protocol requires to broadcast sufficient information about channels to allow for route discovery and find a path from origin to destination of a payment. Additionally, it specifies how can an inermediary collect fees for forwarding payments. Allowing intermediates to collect fees is important to incentive people to forward payments and make an efficient payment network.

2.2 Payment Channel Hubs

Payment hubs leverage the power of Blockhains with Turing complete smart contract languages such as the Ethereum Virtual Machine or EVM, building such a system on a Blockchain like Bitcoin that has a limited scripting language is not possible with the current state of the art. The requirements for building such a payment system are similar to regular Blockchain transactions or with payment channels. It should be completely trust free even towards the hub operator. At any moment the user of the system has full custody of his funds and given some assumptions he cannot lose his funds.

2.2.1 Architecture

A payment hub such as NOCUST[39] consists of an on-chain part, a smart-contract and an off-chain, part the hub server or operator. The smart contract and the hub server are complementary and work together. A user that wish to transact through the payment hub is first required to deposit some funds in the smart contract thus making an on-chain transaction. He then needs to interact with the off-chain hub server. Similarly to a payment channel, the initial action of depositing funds in the Smart contract requires the gas fees to be paid. However, future transactions require to interact exclusively with the

off-chain hub server thus bypassing the need to pay network fees. The smart contract is also responsible for the withdrawal of user funds. Therefore, if the the hub server disappears or does not respond, the user is still able to withdraw his funds through the smart contract living on the blockchain.

2.2.2 Periodic commitment

The hub operator or hub server has access to exclusive functionalities on the contract. He can and should submit a periodic commitment, also called checkpoint. The smart contract will be waiting, at a given fix interval of blocks, for the operator to submit a commitment. The interval between 2 commitments is called a round. The commitment consist of a Merkle root of an augmented Merkle tree where the leaf nodes are the account balances of the users of the hub. This Merkle tree is at the core of the payment hub security. It allows to cheaply and quickly verify the integrity of the user account balances within the tree. The smart contract is used to verify the account state. If the state is invalid or any inconsistencies are detected the smart contract will enter in recovery mode. No further commitments from the hub operator are allowed and the last valid commitment is considered final. The only operation permitted is to allow users to withdraw their funds. The users should monitor the contract state and should stop making off-chain transactions with the operator if the contract enters in recovery mode.
Under the assumption that the user correctly monitors and verifies the contract state, his funds are not at risk. The hub operator is strongly incentivised to not cheat. If he does, victims can initiate a dispute procedure on the smart contract to shutdown the system and therefore the operator cannot facilitate off-chain payments anymore.
Disputes and verifications procedures happen in a smart contract, therefore, we need a computationally efficient way to prove the correctness of an account state. In NOCUST[39] an augmented Merkle tree is used. A value is added to each node in the tree to be hash together with the child nodes. This value consists of the sum of the values of the child nodes. For a leaf node, this value is equal to the balance of the account. Figure **??** shows a simplified representation of the account state Merkle tree. Each leaf node is a hash of the account address, its balance and the root of a Merkle tree of off-chain transactions that occurred during the last round. Note that $transfer_k$ is the root of a regular Merkle tree of transactions approved by user k.
Given such a structure, if the hub operator tries to cheat by increasing or decreasing the balance of one account, the balance of the root node will not be matching the balance of the contract. If the hub operator cheats, there is always at least one inconsistency in the state Merkle tree that a user can dispute.

2.2.3 Challenge mechanism

To guarantee the security of a NOCUST instance, each user should verify locally the integrity of his balance once per round. The verification consists of

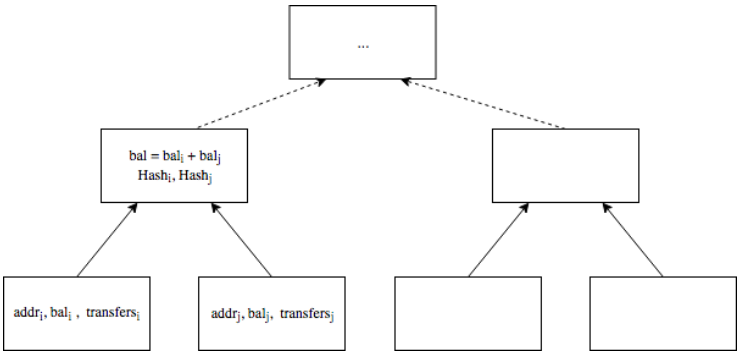


Figure 1: Simplified representation of the augmented account state Merkle tree. The Merkle root at the top will be committed periodically on the Blockchain. bal_i is the balance of account i, $addr_i$ is the Blockchain address of account i, $transfer_i$ is the Merkle root of a tree of transactions that occurred during the round preceding the commitment.

membership verification of the updated account in the state Merkle tree given the root hash of the last commitment on the smart contract. The hub operator should provide to its users the Merkle proof of membership off chain. If the hub operator refuses to send the proof of membership or if the provided proof happens to be incorrect, the user should immediately initiate state update dispute through a specific function on the smart contract. The open dispute will force the hub operator to reply with a correct Merkle proof of membership that matches the Merkle root of the last commitment. If the hub operator fails to answer within a period of time, the hub is shutdown as the contract enters in recovery mode where the only possible operation is to withdraw funds based on the last successful commitment. A round is defined as the fix duration in number of blocks between 2 commitments. A commitment is challengeable during the round following its submission. After this period, if all the challenges were answered correctly, the commitment is considered correct and final and cannot be challenged anymore.
There are two types a of challenges:

State update challenge: It verifies that the balance was correctly updated according to the off-chain transactions made in the previous round. It forces the hub to provide a correct Merkle proof for the leaf account in the tree.

Transaction delivery challenge: It verifies that an off-chain transfer was correctly included in the recipient transaction Merkle tree. It forces the hub to provide the Merkle proof of membership of the recipient account in state tree and the Merkle proof of membership of the challenged transfer in the recipient transfer tree.

If the operator fails to answer any of these challenges after a period of time the

smart contract will automatically enter in recovery mode and the payment hub stop operating.

2.2.4 Transaction finality

An off-chain transaction is considered final and fully confirmed once its root hash has been committed and its dispute period has passed. It means that when making a transfer you should wait for 2 commitments for the full finality. If a round period is 12h then the confirmation time of a the transaction would be between 12h and 24h. To circumvent this long confirmation time and have an instant finality, the hub operator can provide some additional collateral to the smart contract as a guarantee for the transactions that did not pass yet the 2 required commitments. The collateral is used to back the transactions received during the current rounds and the previous round. The amount of collateral needed is equal to the transaction volume over the last and current round. Contrary to payment channels where the collateral is split for every participant and can be used only within a given channel, in a payment hub the collateral can be used freely between all participants without need of rebalancing. The only requirement is that its amount should be inferior to the transaction volume over a period of time. When the hub enters in recovery mode, there is a separate contract function to recover funds from transactions that happened after the last valid commitment.

2.2.5 Payment hub networks

Compared to solutions such as the Lightning network, a single payment hub is very centralised from an architectural perspective. Despite the fact that one does not need to trust the operator for not stealing your funds, it is still a single entity facilitating and authorising all payments. The disadvantage compared to regular on-chain transactions are that the hub is a single point of failure and it is not censorship resistant: the hub can block some users from making off-chain transactions. To respond to these issues several hubs can be used connected using regular payment channels. Routing would be much simpler than in a payment channel network as one would only have several large hubs. The requirement for monitoring the channel state is less of an issue for a hub running on a server that is supposed to be always online. In this context we can assume a much higher reliability between the participants of this payment channel network of hubs rather than when the participants are end user devices such as a mobile phone. Solutions such as REVIVE[38] can be more easily considered to address the issue of exhausted channels. REVIVE allows fast off-chain channel rebalancing.

3 Lightning Network

The Lightning Network is the first and most advanced payment channel network implemented. At the moment, it operates on the top of the Bitcoin net-

work. Its first developments started in 2015 with the Lightning Network original publication[47]. In May 2017 was deployed the Segregated Witness update on the Bitcoin protocol. This update made the Lightning Network possible and the first implementations started to operate on the Bitcoin test network. In March 2018 the first implementations for the Bitcoin main network were released and the first nodes and channels where setup. In August 2018 the Lightning Network is still under intense development. The total value involve is currently about 78.9 BTC or about 639'000 USD in August 2018.

3.1 Implementations

The Lightning Network is an open source project involving hundreds of developers around the world. Its key components are the BOLT specifications[15] that aim at standardising the base protocol. There is currently 3 major implementations of the BOLT standard that allow to run a lightning node:

LND[41]: Developed by lightning labs written in the Go language. It supports the alternative Bitcoin client btcd[48] and it is noticeably the only one to support a Bitcoin light client, neutrino[40], as blockchain source.

c-lightning[30]: Developed by Blockstream, written in the C language.

Eclair[28]: Developed by ACINQ and written with the Scala language.

Each of these implementations provides a command line interface with full wallet functionalities to transact on the Lightning Network. They generally require to be connected to a Bitcoin full node such as the Bitcoind[**?**] client, also called Bitcoin core. With a Lightning network client one can keep track of the network graph, open and close channels, send transactions and forward payments.

3.2 Network statistics

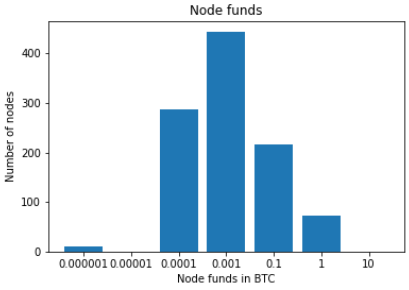
3.2.1 Data acquisition

During the course of this project a Lightning node was setup using the LND implementation. The node was run with a BTCD Bitcoin full node[48]. It was taking a snapshot of the network every hour during 3 months. The data presented in the section is from a snapshot of the network from late August 2018.

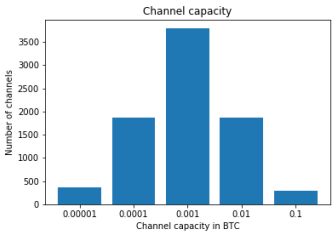
3.2.2 General network statistics

The Lightning Network currently has 2496 nodes of which 1449 are considered active with at least one open channel. A node has an average capacity of 0.054 BTC (442 USD) with a median of 0.0038 BTC (31.2 USD). The node capacity is defined as the sum of the capacities of its channels. Figure 7.a shows the distribution of node capacities rounded to the order of magnitude. There is currently 7915 active channels, the average channel capacity is 0.0096 BTC (79

USD) and the median is 0.002 (16.5 USD). Figure 7.b shows the distribution of channel capacities. One can notice the large difference between median and average of these 2 metrics.
The network has been growing very fast, from 61 channels in January 2018[2] to about 7915 today. Figure 3 shows a representation of the network graph as of August 2018. We can quickly notice some well connected nodes in the center with many channels open that are probably payment hubs. Nodes in the periphery with one or two channels are certainly more end user wallets that do not forward any payments.



(a) Node capacity



(b) Channel capacity

Figure 2: Distribution of node capacity (Sum of the capacity of all its open channels) and channel capacity

The data was captured by our node for a period of 3 months. 3 months represent half of the current life span of the Lightning network on the main Bitcoin network that was officially in March. Figure 4 shows how the network has evolved during the course of this project. The network is currently growing at a steady rate of 300 new nodes per month and about 2000 new channels per month. The total network capacity is also growing. We can notice some radical spikes and drops in the total network capacity because some nodes are sometimes joining and leaving the network with massive amount of funds[3]
The reader should note that these statistics do not include private nodes as they are difficult to detect. However this number is most likely a minority of the nodes as it is an experimental feature with limited usage. (See section 3.3.2)

3.2.3 Network topology

In this section the topology of the network is studied in depth. On the first place one looks at the number of hops needed to join 2 random nodes. In Lightning, we want to do multi-hop payments such as sending 1 BTC from Alice

[2] https://twitter.com/lopp/status/1022077129771687936

[3] User injecting about 40 BTC in the Lightning Network for a period of 2 weeks. `https://medium.com/andreas-tries-blockchain/bitcoin-lightning-network-2-we-must-first-become-the-lightning-network-49c46953c1d7`

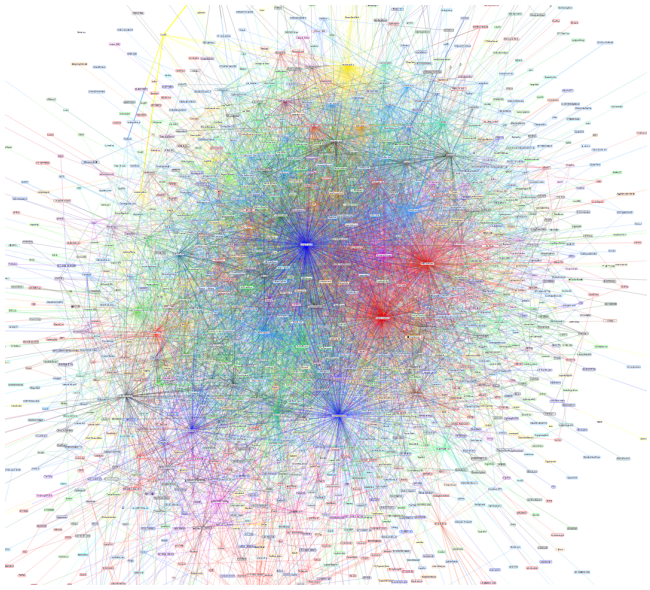


Figure 3: Visualisation of the lightning Network graph. Source[14].

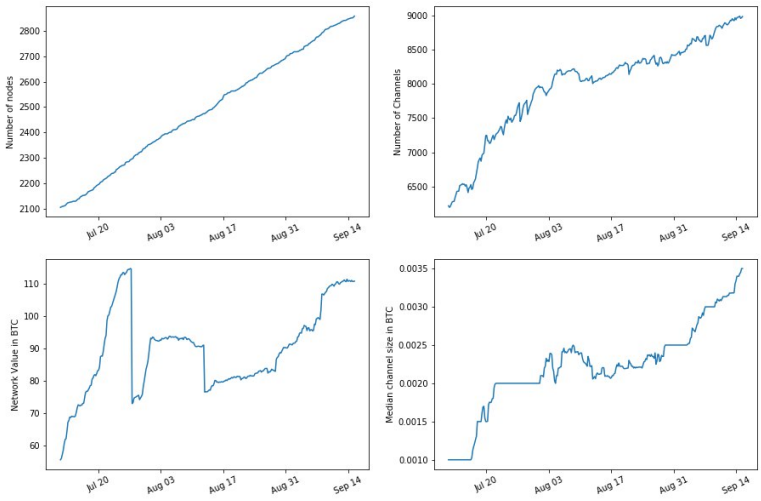


Figure 4: Evolution overtime of the number of nodes, number of channels, the total network capacity and the median channel size.

to Dave through Bob and Carol: `Alice -> Bob -> Carol -> Dave`. Dijkstra algorithm[32] was used for finding the shortest paths between nodes in a graph. It requires on average 2.3 hops or routing nodes to join 2 random nodes. This would result on an average of 3.3 linked transactions to send a payment between 2 random nodes. Figure 5 shows the probability distribution of the number of hops. From the graph we can conclude that most of the time up to 3 routing nodes is sufficient to link any sender to any recipient.

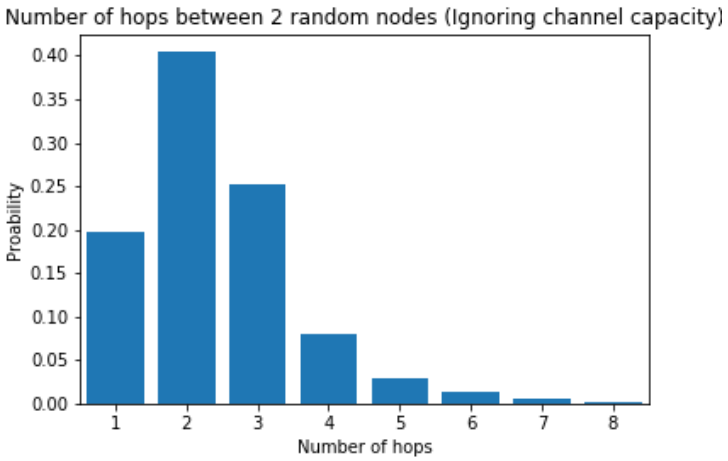


Figure 5: Probability distribution of the number of hops (intermediary nodes) required to join random nodes in the network.

Interestingly, the shortest path between 2 random node has dropped over time. When we started capturing the data in July, the average number of hops was about 2.6, while now it is around 2.3 nodes. Figure 6 shows the evolution of the average shortest path.
Figure 3 shows that large hub nodes tend to appear in the network. There are multiple reasons for having large hubs in the network. Nodes need to be reliable, always responsive and need to provide a large sum of collateral for an efficient routing. Additionally, at the moment, it is in practice very difficult to open a channel with a random node because most of them are not configured to accept channel openings.
Table 1 shows the top 10 nodes with the most channels. They control 3001 channels or 36.6% of all the channels. Additionally, more than half of the total network capacity, 55.5%, lies in their channels. Similarly to the mining centralisation of the Bitcoin hash power, these numbers show also a high level of centralisation in the Lightning Network topology. Out of the top 10 nodes, 7 are hosted in well known cloud computing providers such as Amazon Web Service, Google Cloud Platform and Digital Ocean.

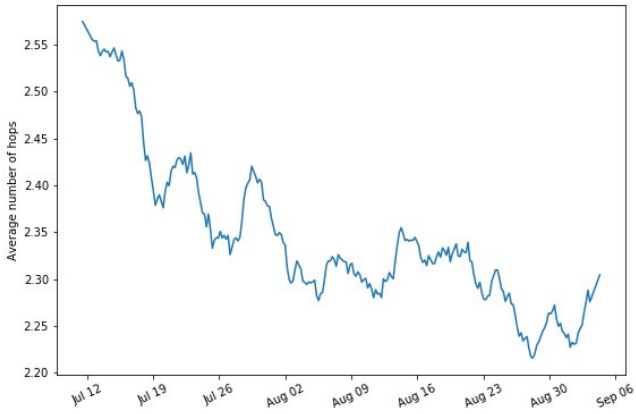


Figure 6: Evolution of the average shortest path between 2 random nodes over time.

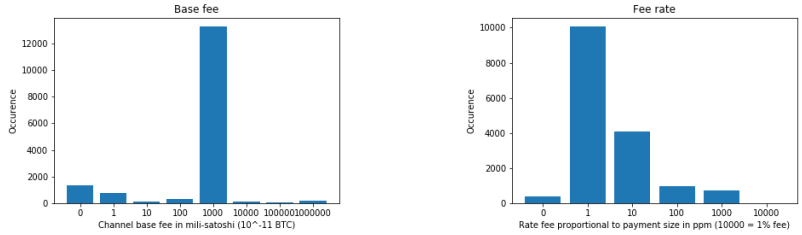
3.2.4 Network fees

When opening a channel each party has to chose his fee policy. This fee will be collected by the channels owners if payments are forwarded through them. There are 2 fee policies per channel, one for each of the directions. The fee policy is publicly advertised and can be updated after the channel opening. Fees are different from regular on-chain transactions, where the fee is proportional to the size in bytes of the transaction. In Lightning, the fee is composed of a fixed fee called the base fee, plus a fee proportional to the payment value. The proportional part is called the fee rate. As routing nodes have to lockup a sum of collateral to back transfers, it is coherent to charge users proportionally to the use of this collateral.
Current Lightning network fees are extremely low. A 10 USD payment through a random link will cost, according to the average fee policy, 1.12 Satoshi or $1.12*10^{-}8$ BTC or about 0.000067 USD. However, in practice the fees can easily be an order of magnitude more expensive. When sending money to popular merchants the fees will be higher because their routes are in high demand. Figure 7a shows the base fee distribution in Satoshi and figure 7b shows fee ratio distribution in part-per-million. The median fee policy reflects the default configuration in the Lightning clients: 1000 Satoshi base fee and 1 ppm rate fee. It is very good news for users to have very low fees. However, if the fees stay so low it will be hard to incentivise people to operate routing nodes. Some operators have reported[4] earning below 0.5 USD per week with more than 7000

[4]https://twitter.com/alexbosworth/status/1019985943321706496
https://medium.com/andreas-tries-blockchain/bitcoin-lightning-network-2-we-must-first-become-the-lightning-network-49c46953c1d7

Rank	Alias	Nbr. of Channels	Capacity BTC	Capacity USD	percent of network value
1	tady je slushovo	524	4.352	35690	5.52%
2	TrueVision.club	423	4.449	36483	5.6%
3	rompert.com	410	2.375	19472	3.01%
4	TheLND.com	277	0.168	1378	0.21%
5	fairly.cheap	277	22.279	182685	28.24%
6	KRYPTO.KOELN	275	0.589	4831	0.75%
7	Bitrefill.com	238	4.572	37488	5.79%
8	mainnet.yalls.org	204	3.028	24827	3.84%
9	ThrobbingSausage	194	1.968	16136	2.49%
10	AY_YILDIZ_LN	179	0.077	627	0.1%
Total		3001	43.857	359627	55.5%

Table 1: Top 10 nodes with the most open channels.



(a) Base fee

(b) Fee rate

Figure 7: Fee policy parameters distribution of channels

USD at stake locked in their channels. The Lightning network is still in an experimental stage, probably no participant expects profitability at the moment.

3.3 Payment Routing

Reliably routing payments in the Lightning Network is one of the major challenges. Its developers originally claimed to have very strong privacy properties: Channels can be private and isolated between 2 parties with no broadcast ever needed to take place. However, this conflicts with how to do multi-hop routing. How to find a route if channels are kept private? Routing payment across several links currently has a high rate of failure. The reasons are multiple: non existing route, outdated network graph, node not responding and unbalanced channels.

3.3.1 Routing requirements

3.3.1.1 Beforehand known routing

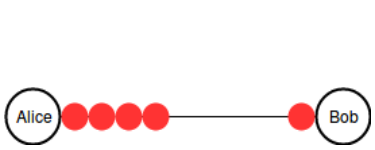
The Lightning network is often compared to the internet network and how a similar routing approach can be used. However their nature is very different. When sending an IPv4 or IPv6 packet on the internet, their is no need to calculate an exact route prior sending the packet. The packet is successively sent to routers that maintain routing tables updated by protocols such as BGP. In fact you do not even know the route that your packet has taken to reach his destination. In Lightning, our packet is an atomic set of transactions cryptographically signed to create a linked payment. Therefore, the sender authorising the transaction needs to know the exact route with its parameters, such as channel fees and capacities. If any of these parameters is outdated, or if any of the intermediate nodes on the route goes offline, the payment will fail. Therefore, in practice the sender needs to have a complete and as up to date as possible copy of the full network graph to create a reliable route. In the next section we will discuss the scalability issue caused by the requirement of maintaining a local network graph.

3.3.1.2 Dynamic network graph

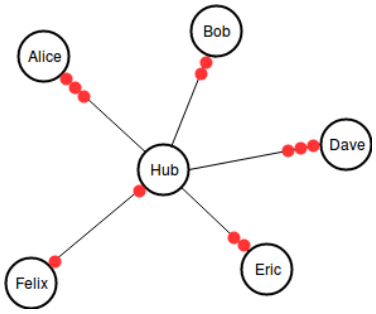
A second major difference with the internet network is that the internet network routing works because the graph is relatively static. The edges of the network are the "cables", internet lines do not change very often. Lines are sometimes closed and new lines are added, but overall it has a low impact on routing. In a payment channel network, every single transaction changes the network state. A payment from A to B decreases the capacity in the direction A to B and increases the capacity in the direction B to A. A naive approach for a payment channel network can be to broadcast every transaction to allow others to correctly update the balances in the network graph. This approach is obviously not scalable as the number of messages exchanged grows in a quadratic fashion with the number of users making transactions. The Lighting Network took a different approach: for scalability and privacy reasons it was chosen to not broadcast transactions. This choice reduces the number of messages needed to be exchanged to maintain the network graph, however it makes routing more difficult. We will look at this issue in the next section.

3.3.1.3 Exhaust channels

The design choice of the last section leaves us with the impossibility to know channel balances of channels that you do not control. You can see from the original on-chain funding transaction the total capacity of the channel but you cannot know what is the current repatriation of these funds. If Bob and Alice have an open channel between them with a capacity of 5 BTC you cannot know how much of this 5 BTC belongs to Bob and how much belongs to Alice. This makes routing more difficult: if Alice owns 4 BTC and Bob 1 BTC, if someone



(a) Single channel with capacity of one from Bob to Alice



(b) Situation where transfer are limited to sending one unit to Felix.

Figure 8: Illustration of channel capacity.

tries to route a payment of 2 BTC through Alice and Bob in the direction Bob to Alice, the payment will fail as Alice does not have the capacity to forward this payment. In this example the advertised channel capacity was 5 BTC but in reality it is of 4 BTC in one direction and 1 BTC in the other. Figure 8a illustrates this situation, where fund in the channel are represented by red circles.

In reality, users like Alice and Bob will open channels with a Lightning hub. To facilitate any payment, the hub needs to provide collaterals. Figure 9c illustrates a situation where the possible transactions are very limited. The only possible transaction is to send 1 coin to Felix, all the other participants cannot receive any funds. These situations are very common in practice, where opening a channel with a hub will most likely not provide spontaneously collaterals in your channel. You need first to spend some of your funds to increase your inbound capacity. The limit in inbound capacity or receivablility of funds is currently a major limitation for the Lightning network. A possible solution would be to rebalance the channels of the hub. However, it requires to pay on-chain transaction fees. Off-chain rebalancing solutions such as REVIVE[38] were proposed. althouguh it is not suitable for Bitcoin's limited scripting language.

Another option to address the issue of Exhaust channels is to adjust the fee policy of the channel. If the channel is Exhaust in one direction, the node can broadcast a fee policy update to reduce the fee in the other direction to favour the rebalancing. Fees can even be negative to pay for a rebalancing. The biggest drawback of this approach is that policy updates have to be broadcasted to the whole network. Therefore, the number of policy updates to keep track grows quadratically with the number of users. Currently, this feature is being used by a minority of the nodes. We measured that less than 5% of the channels ever updated their fee policy.

A last approach to solve this liquidity requirement issues is to assume a certain level of trust in a hub operator and to relax the collateral requirements by allowing transactions to be backed by collateral located in a different channel of

the hub. This is the approach taken by Fairlayer with its Extended Lightning Network[5].

3.3.2 Privacy

When opening a channel with clients such as LND, the default option will be to create a public channel. This means that the opening will be broadcast and advertised in the network. The channel will be visible in the network graph and routing is possible. If the channel is kept private it will not be publicly advertised, therefore, incoming payments are difficult as the sender needs to obtain the routing information via a different communication channel. A private node is a node with exclusively private channels, some mobile wallets such as the Bitcoin Lightning Wallet are configured to be private nodes. They only open private channels to not expose any information such as Bitcoin public keys and IP addresses. However, incoming payments are either disabled or require routing hints. Routing hints indicate to the sender what intermediary public node should the payment go through to reach the final destination. This hint is included in a payment request and often shared in the form of a QR code.

3.4 Autopilot

Autopilot attempts to simplify the use of the Lightning Network for end users. It automates some complex tasks needed when operating a Lightning node. It is proposed as a module for the LND implementation[6]. Autopilot automatically manages the channels of the node according to a set of heuristics fully configurable. It is implemented as a simple closed control loop that takes as input parameters such as wallet balance, channel opening, channel closing and channel updates. At each change in the inputs it is evaluated if it needs to open a new channel with how much funds and to which node to connect to. The default heuristic, given a minimum channel size, maximum channel size and percentage of the wallet funds to allocate, will open channels to specific nodes chosen using a preferential attachment following the Barabási–Albert model[29]. This model allows the generation of scale-free networks. A scale-free network is a graph with the property that the number of links k originating from a given node exhibits a power law distribution $P(k) \sim k^{-\gamma}$ of parameter γ. A scale-free network can be constructed by progressively adding nodes to an existing network and introducing links to existing nodes with preferential attachment so that the probability of linking to a given node i is proportional to the number of existing links k_i that a node has. Additionally scale-free networks have the interesting properties of being relatively robust to network failures and have a low average distance between 2 nodes.

[5] XLN: `https://medium.com/fairlayer/xln-extended-lightning-network-80fa7acf80f3`

[6] Autopilot: `https://github.com/lightningnetwork/lnd/commit/306c4aef8e3af44fb3f2d8f52fc887f2c48e9c04\#diff-ecf4f0aacb1b1749d365ca301f17036e`

The goal of Autopilot is double. It create a smoother user experience by automating complex tasks, and it drives the network topology towards a more scalable and reliable structure.

3.5 Watchtowers

Watchtowers attempt to solve the problem of the constant online presence required to protect a payment channel from other parties to potentially steal funds. If a party chose to force close a channel with an outdated state in his favour, the other party should immediately respond to the fraud by submitting the corresponding hash pre-image that allows him to cancel the operation and get his funds back. When force closing a channel in Lightning, a slack period is given to allow the closure to be mitigated. This slack period is a channel parameter chosen during the channel opening, usually set to a value between 14 and 144 Bitcoin blocks (between 3.5 hours and 1.5 day). If a wallet node such as a mobile phone is offline for a period of time above this duration, the attack is possible. To solve the issue, one can rely on a third party service provider that monitors the channel and is willing to submit the mitigation transaction if needed on the behalf of someone else. Note that this third party does not have any private key that allows him to control the funds in the channel, he should just have the hash pre-image that allows to mitigate a fraudulent closure. However the user needs to trust the watchtower for effectively reacting to an attack. More elaborated approaches were proposed, such as PISA[42], to reduce this risk. However, it is only suitable for blockchains with more advanced Smart contract capabilities such as Ethereum.
In Lighting, the protocol to define the interactions between a node and a watchtower is currently being drafted[7]. Wallets such as the android Lightning Wallet[8] provide their own watchtower service directly integrated. As a simple workaround, many Lightning wallets aiming to be user facing do not allow incoming Lighting transactions. By not allowing incoming transactions the user is not put at risk of being victim of an old state submission. With only outgoing transaction, the most recent state is always advantageous for the other party.

3.6 Payment channels Wallets

Requirements of current implementations of payment channel networks make the user experience difficult. We first review the current challenges for users, then we will present existing wallets that focus on user experience.

3.6.1 Challenges

In the previous sections we explained how the Lightning Network operates. However some requirements are problematic to user experience. The challenges are reviewed here.

[7] Wtwire WIP: https://github.com/lightningnetwork/lnd/pull/1512
[8] Lightning Wallet: http://lightning-wallet.com/

Network graph : To enable routing, each wallet needs to construct the full network graph. In august 2018 the graph weighted about 7,7 Mega byte with 2000 nodes and 8000 edges. The size is currently acceptable. However, Lightning Network is expected to scale beyond the current Bitcoin user base. Bitcoin currently has 22 millions Wallets and between 2.9 and 5.8 million active users[12]. The Visa payment network has 1.07 billion active users and a total of 3.19 billions credit cards[37]. Lightning is expected to grow by 3 orders of magnitude from its current state to reach Bitcoin scale and 3 more order of magnitude to reach Visa scale. Obviously, if the graph has to grow linearly with the user base it will not be sustainable for a wallet to operate on devices such as a smart phones. Additionally the graph needs to be constantly updated and synced at all times to guarantee reliable routing. Any channel update has to be broadcasted to all participants. Therefore, the number of exchanged messages grows quadratically. If any parameter from the graph for a payment is outdated, the payment will fail. For these reasons the wallet node will have a high bandwidth consumption and the initial sync can take up to an hour.

Online presence : A wallet should constantly monitor the Blockchain to protect his funds from being stolen and cannot go offline for an extended period of time. This is the issue that Watchtowers attempt to solve. In addition, online presence is not only required for security reasons but also to received funds. The receiving party is required to sign the HTLC to receive funds. This contrasts with regular on-chain transactions where no interactions are required for the receiver.

Blockchain source : A lighting node requires a Blockchain source to operate. It is needed first to authenticate channels announcements. When receiving these announcements one has to make sure that these channels exist through verifying the corresponding UTXO of the funding transaction. If this verification is not done the node can potentially accept junk channels and he is exposed to obvious denial of service attacks. The Blockchain source is required also to monitor the submission of potentially fraudulent forced closure. Last, it is used to submit channel opening and closing transactions. Mobile clients, obviously cannot afford to run a Bitcoin full node as they lack sufficient hardware resources. Regular SPV nodes or light client node verifying only the Bitcoin block headers do not fulfil the security requirements to operate Lightning node. The reason is that a SPV client cannot guarantee that a transaction was not included in the Blockchain. To solve this issue new Bitcoin light clients are begin developed such as the Neutrino client[40]. However this client is still experimental and is not ready to operate on the Bitcoin mainnet.

Inbound capacity : When opening and funding channels, initially, the inbound capacity will most likely be zero. Unless the hub provides you with

some collateral, you will never be able to receive more funds than what you have already spent.

Channel management : The wallet is required to open, update and close channels. These complex tasks are a burden for users. This can be done automatically by an Autopilot feature described in section 3.4. However, it is hard to completely abstract to user the channel management. Having several channels means having several inbound and outbound capacities, on per channel. Maximum payment size is equal to the largest channel capacity. Therefore one gets unpredictable and recurrent fees for the on-chain operations (for instance, funds can be lockup). It is hard to hide all the complexity because the users would have the feeling of not having full control of their funds.

Payment failure : Payment are often failing for the reasons described in section 3.3.1. One reason is the impossibility to tell with certainty if the channel we want to use is exhausted in a given direction or not. The only solution is to make a guess and have a try. Additionally some parameters from the network graph might be outdated, thus making the transaction invalid. Nodes along the payment path might not be responding or failing. Finally, a route might simply not exist

Funds backup : To backup a regular wallet, users are often asked to write down a secret such as a 12 or 24 words passphrase from which the Bitcoin private key is derived. With this passphrase users can recover their funds if they lose access to the wallet. However the passphrase is not sufficient to recover the funds in the lightning channels. The reason is that the channels keys are updated after each payments. If the user looses his mobile phone on which the wallet was installed he would not be able to recover the funds currently in the channels.

To address these issues current wallets have adopted drastic trade-offs. To solve the online presence issue and the Inbound capacity issue some wallets are simply not allowing incoming payments making the off-chain wallet unidirectional. By only making outgoing payments you are not exposed to the risk of the other party submitting on old state as the balance would always be in your favour. Additionally there is no need to confuse users with inbound capacity limitations and hub-side collateral. A wallet that only allows outgoing off-chain payment and only on-chain incoming payment is obviously not a fully functional off-chain wallet.
The second important trade-off that some wallets have taken to solve the Network graph issue is to rely on a third party route provider. Instead of generating the route for a payment locally, the wallet will query a route server taking care of maintaining the graph. This releases the wallet from having to synchronise the network graph, which is very resource consuming. However, this route provider is a single point of failure for the wallet and requires a certain degree of trust in the route provider for giving the optimal route in terms of fees. In a

near future, when the Lightning Network fees will be higher, we could imagine an attack where the route provider could make the payment going through his channels with higher fees.
There are currently on the Lightning Network about 5 popular merchants such as Satoshi Place[9] and the Blockstream Store[10]. One can make payments to these merchants with a relatively high chance of success. As there are very few entities accepting payments, it is fairly easy for routing nodes to artificially provide collaterals towards these entities. Making a payment between 2 regular wallets is much more uncertain and it is often impossible to find a working route.

3.6.2 Existing wallets

We will present here some Lightning wallets with a focus on user experience. Note that most wallets presented are based on the LND implementation and are open source.

3.6.2.1 Mobile

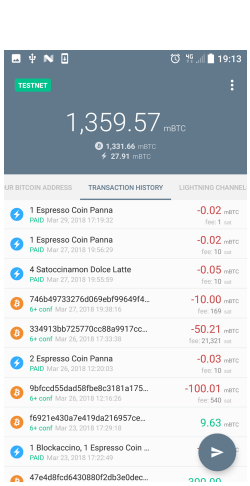
Mobile wallets have the challenge of being very restricted in hardware resources. These devices are also very likely to go offline, sometimes for a long period of time.

Eclair wallet mobile Developed by ACINQ, it uses their own Lightning client. It is a unidirectional wallet allowing only to send funds, channels have to be managed manually. This wallet addresses the issue of funds backups by connecting to the Google Drive account of the user and synchronising the most recent channels keys. The keys are encrypted using the static wallet passphrase before being uploaded. Eclair wallet does not do any local Blockchain source and it is not authenticating channel announcements properly[11]. It makes the wallet vulnerable to major denials of service attacks as described in the previous section. Figure 9a shows a screenshot of the wallet.

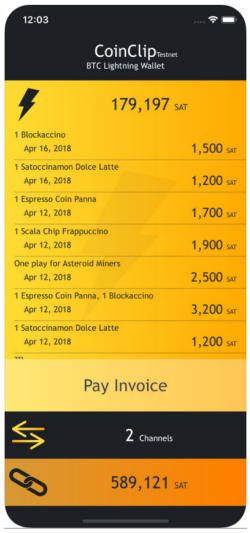
CoinClip Mobile wallet for IOS, currently only on the bitcoin testnet. CoinClip has the specificity of being the only wallet to open public channels. Each wallet is visible on the network graph and is clearly identifiable. CoinClip is also unidirectional, it does not offer any possibilities to backup channel funds. It is the only closed source wallet presented here. Figure 9b shows a screenshot of the wallet.

Bitcoin Lightning wallet Bitcoin Lightning wallet is the only mobile bidirectional Lightning wallet. It works together with the Olympus server [20], fulfilling several tasks. It backs up the channels keys allowing to recover

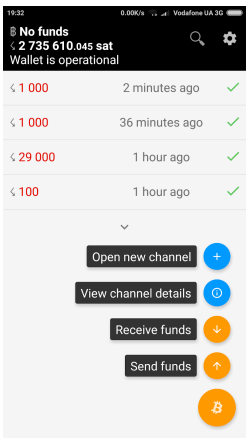
[9] `https://satoshis.place/`
[10] `https://store.blockstream.com/`
[11] Eclair's channel authentication issue: `https://github.com/ACINQ/eclair-wallet/issues/101\#issuecomment-408618910`



(a) Eclair wallet



(b) CoinCplip wallet



(c) Bitcoin Lightning Wall

Figure 9: Lightning mobile wallets

channel funds. As it is a bidirectional wallet, there is a need to monitor the Blockchain for outdated state submission. The Olympus server fulfils this role and acts as a pre-configured watchtower. Last, this wallet relies on a route provider, it does not fetch the network graph itself. This wallet solved most of the challenges through relying on third parties. It assumes a higher level of trust in the operator of the Olympus server. The server operator can easily setup attacks to steal funds in the wallet as it controls the Routing, the watchtower and the remote blockchain source. Figure 9 shows a screenshot of the wallet.

3.6.2.2 Desktop

LND wallet : This is the command line tool of the LND implementation, it is a fully functional wallet holding private keys. It offers all the features that one can expect from a Lightning wallet: it is bidirectional, it constructs fully the network graph and has autopilot embedded. The major drawbacks are that it requires a Bitcoin full node to operate on the main network. It has to be used with care as it does not provide any mechanism to ensure security when the wallet is offline. It is not meant for everyday use but rather as a development tool for advanced users. It is suitable to run on servers to setup routing nodes.

Zap wallet : Zap is developed by independent developers. It is the the most popular desktop wallet providing a graphical, minimalist, clear and intuitive interface. It is written in JavaScript, using Electron. It uses the

Neutrino light node despite its experimental stage. It does not provide any watchtower features neither channel backups.

htlc.me : It is a popular web-based lightning wallet very simple to use and get started. However, it is custodial, the private keys are stored on their server. Full trust is required in its operator.

3.6.3 Comparison

Table 2 is a summery of the different characteristics of the wallets presented before.

	Platform	Mainnet	Bidirectional	Autopilot	Authenticated channels	Offline safe	Channel backups	Blockchain source	LN graph
Eclair mobile	Android	✓	✗	✗	✗	✓	✓	Remote	Local
CoinClip	IOS	✗	✗	✗	?	✓	✗	?	Local
Bitcoin Lightning wallet	Android	✓	✓	✗	N.A	✓	✓	Remote	Remote
Zap desktop	Desktop	✗	✓	✓	✓	✗	✗	Neutrino	Local
LND	Desktop	✓	✓	✓	✓	✗	✗	Full node	Local

Table 2: Lightning wallet comparison

3.7 Other payment channels network implementations

The Lightning network, despite being the first and currently the most advanced payment channel network, it is not the only one. We will quickly present here some projects developed for Ethereum.

Raiden [22]: It is the most well know payment channel network currently in development and running on a test network. As Raiden operates on Ethereum, that provides a Turing complete programming language, it can benefit from amelioration such as PISA[42] or Sprites[43].

Celer [7]: Focuses on state channels or general computational purposes rather than just payments. Their routing algorithm DRB is inspired from the Back pressure algorithm[45] originally designed for wireless networks.

SpankChain [25]: Payment channel solution for Ethereum developed for the adult industry with a strong focus on the privacy features that payment channels can offer. SpankChain payment hubs are based on Perun[33] virtual channels.

Trinity [26]: A payment channel network solution for the NEO Blockchain[19].

4 Liquidity payment hub

A first NOCUST payment hub was deployed on the Ropsten Ethereum Test network in March 2018. After a period of testing, a payment hub was deployed on the Ethereum main network[12] in late June 2018. It currently handles a low amount of Ether: about 6.67 USD worth Ether is in the system. Because the project is in early stages and the priorities are to ensure reliability and security, the deposits are limited to 0.001 Ether (0.28 USD). In the current deployment a round lasts 4320 blocks or about 18 hours and in early September 2018 the contract was in its 61st round. The current version does not provide any collateral and therefore 2 commitments are required for the transaction finality. The hub registered 1623 addresses and facilitated 1736 payments as of September 2018.

4.1 Implementation

A NOCUST payment hub infrastructure is composed of 3 major software elements: the on-chain smart contract, the hub server and the client.

Smart contract : The smart contract handles users deposits and authorises withdrawals. It has challenges functions that allow to verify the integrity of an account balance. Last, the smart contract is expecting the periodic commitment from the hub operator.

Hub server : The hub server is controlled by the hub operator. It is responsible to submit the regular commitment into the smart contract. The hub server should reply to all challenges that are posted on the smart contract by its users. The operator needs to approve the off-chain transfer and include them in the commitment. It has to take care that no double spending happens.

Wallet client : The wallet client is needed to operate an off-chain wallet. The client retrieves the Merkle proofs and verifies it against the smart contract state. If the security status of the hub is considered safe, the client allows to create off-chain transfers. In addition, the client enables to make deposits, withdrawals and initiate challenges.

4.1.1 Gas cost evaluation

At each commitment the hub operator submits the root of the account Merkle tree. The verification of membership of an user account within the Merkle tree is at the core of NOCUST hubs security. However, as we are using a smart contract as a recourse mechanism, it costs gas fees. Users should verify first the proof of membership locally. Whenever a fraud is detected, the user should do the verification on-chain by starting a challenge. Note that withdrawals also require to submit a Merkle proof.

[12] Hub smart contract address `0xac8c3D5242b425DE1b86b17E407D8E949D994010`

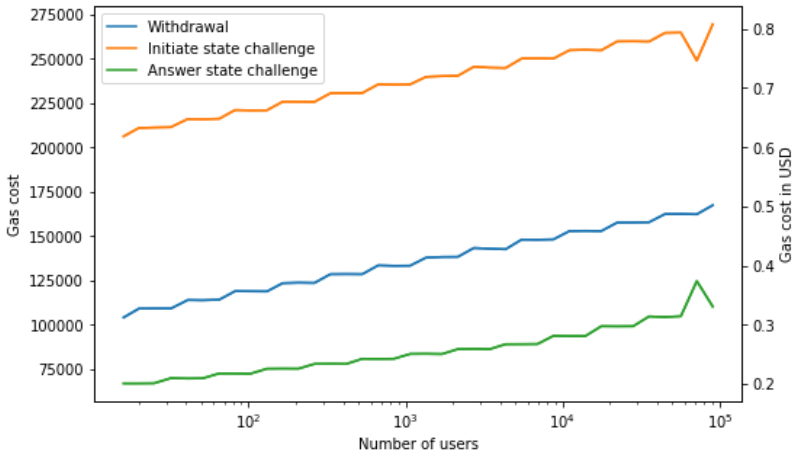


Figure 10: Gas costs of Withdrawals, State challenge initiations and answer to the state challenges depending on the number of users. Gas price used is 10 Gwei and Ether price 300 USD.

The membership verification needs to be practical on-chain, in this section we will look at the gas costs of such operations. The bottleneck of on-chain calculations is often data costs. Data associated to contract calls is permanently stored on the Blockchain. This means that every node in the network will have to keep a copy of it. This is obviously expensive and that translates in high gas costs. Storing a 256 bits word on-chain cost about 20'000 gas[49] at gas prices of 10 Gwei, which means 0.0002 Ether or 0.055 USD at current Ether prices of 300 USD.
With a Merkle tree, the size of the proof grows logarithmically with the number of leafs in the tree. In NOCUST, the number of leafs in the account tree is equal to the number of users. Because we are using an augmented Merkle tree, for each hash value in the path to the leaf we need an extra value for the cumulative balance in the unknown branch. Therefore, the data complexity for verifying a proof is $\mathcal{O}(log(N))$, with N being the number of users. We run instances of the hub with the contract deployed on a private test network to measure the gas cost depending on the number of registered users. Figure **??** shows the results of the experiment. We can notice that the gas costs grows logarithmically with the number of users. The step-wise shape of the curves is due to the addition of a new level in the Merkle tree, meaning that the costs increase only when a new level needs to be added in the Merkle tree.
Note that if the hub is not cheating, it is never required to initiate a challenge. When the hub is being cooperative, it offers a second type of withdrawals. The hub can authorise withdrawals without verifying the account state on-chain, making it cheaper and constant cost. It is still secure because the withdrawal would still have to go through a full challenge period before being delivered.

4.2 Comparison with payment hubs

Payment channel networks like the Lightning network and payment hubs such as NOCUST are both off-chain scaling solutions or layer 2 solutions that build on the top of existing blockchain to improve the overall transaction throughput. We will compare here the two approaches, pointing their advantages and disadvantages.

Online presence requirements : The online presence requirements for both approaches is more constraining than for regular Blockchain transactions. With payment channels you are required to monitor your channels to protect your incoming funds. If not done properly, the other party of the channel could submit a forced channel closure with an outdated state in his favour. Typically, the maximum interval is 24h. It is therefore needed for the user to come online once a day. We have showed in the previous section how this can be mitigated with watchtowers.

With payment hubs, users are required to come online once per round to audit the last commitment of the hub and verify the integrity of their account. If not all the checkpoints are verified, it would allow the hub operator to cheat and manipulate users accounts.

Collateral requirements : A major issue with payment channel networks is the lack of collateral or lack of liquidity. When executing a payment, if one of the intermediary nodes does not have enough collateral, the payment will fail. This worsens if a node on the network, potentially a merchant, receiving a lot of funds draining all the routes in his direction. A node with 10 channels that wants to guarantee the possibility of forwarding funds up to a value of 1 BTC for each channel needs to provide 10 BTC of collateral that will be lockup for a long period of time. The collateral requirements of a payment channel network is very high. Additionally, it should also be taken into account that payments are multi hops that require the same amount of collateral at each hop. In a payment hub such as NOCUST no collateral is required if participants are willing to wait longer for transaction finality. If users want instant transaction finality, the hub has to provide collateral to back the transactions of the current and previous round as these might be reverted in case of a failed challenge. Therefore, the collateral required is equal to the transaction volume over the period of two rounds. Once the round has passed the collateral can be reused for the next round.

In the context of a payment hub network, NOCUST hubs would be connected together with payment channels. Additional liquidity would be needed to operate the channels between the hubs. However, as hubs are designed to support in the hundreds of thousand users there will be much fewer hubs that there are Lightning nodes and therefore much less channels in which to provide liquidity. Routing would mostly happen exclusively directly without any hops, thus reducing the risk of failure due to a lack of liquidities.

Routing : Routing is a major issue in payment channel networks as it scales. Each participant needs to keep a complete copy of the network graph. This approach becomes quickly not suitable for light clients with limited hardware resources. In the context of a payment hub network, routing would be trivial because the number of nodes would be much lower. In addition, all hubs are assumed to be more "professional" entities that are dedicated to build a proper hub setup. In the Lightning Network routing nodes can be user servers and are therefore much less reliable.

Resilience : In a payment hub network, many users rely on a single hub for all their payments it's therefore a single point a failure for many users. From an architecture point of view, payment hub networks are more centralised than a payment channel network like Lightning. Users of payment hub network would need to rely on several payment hub to improve their resilience.

Censorship resistance : The operator of a payment hub can choose who is allowed to transact through him and can block users at his own will. It can get better in a situation where there are several hubs, where a censored user could simply choose another hub to transact. A payment channel network is much better in terms of censorship resistance. If a node refuses to forward a payment, another node can be selected. Due to the use of Onion routing encryption, an intermediary node cannot know the original and final destinations of a payment beyond its directly connected peers.

4.3 Other payment channels hubs

A similar idea to NOCUST is Plasma[46]. Plasma is considered as one of the major scalability solutions for Ethereum. It is developed by several startups such as OmiseGo[21] or Loom Network[17]. The major difference between Plasma and NOCUST is that Plasma is based on Unspent Transaction Output (UTXO), while NOCUST is account-based. Therefore, users in Plasma need to monitor and challenge UTXOs rather than a unique user balance. Plasma focuses more on general computational purposes than just payments. Plasma comes in several forms, we will discuss them in the next sections.

4.3.1 Plasma cash

Plasma cash is a Plasma implementation aimed for tokens of the ERC721[34] standard. These tokens are non-fungible, each token is unique and therefore distinguishable from each other. These tokens are also called collectible.
Plasma cash uses a sparse Merkle tree[31]. Sparse Merkle trees are similar to regular Merkle trees, being their main difference that the leafs of the tree is the complete output space of a given hash function. Therefore, a major part of tree will be empty and many leafs will be equal to the empty element. Sparse Merkle trees are less efficient than regular Merkle trees, but their advantage is that they allow for an efficient proof of non-membership in the tree.

This data structure is efficient against double-spending, it allows for a straightforward verification of if an UTXO was already spent or not. There is a trade-off between flexibility and fungibility. In Plasma Cash, tokens have to be spent in the exact denomination in which they were created. There is no way to spend less than or a fraction of a specific output.

4.3.2 Plasma debit

Plasma debit similarly uses Sparse Merkle trees but the leaves are transactions and updated account balances. It gives flexibility and outputs can be freely separated. Users are required to open a regular payment channel with the hub operator. Therefore, it requires to pay the gas fees for the opening transaction. To withdraw, the user claims the payment channel and a challenge period starts for the submitted account state.

5 Liquidity off-chain Wallet

I have developed the first mobile wallet for the Liquidity Network payment hub based on NOCUST[39]. It is a fully functional wallet that secures the private keys and that allows to make transactions off-chain through a NOCUST payment hub and regular on-chain transactions on the Ethereum main network. It is available for Android on the Google playstore[13] and for IOS[14] on Apple Appstore.

5.0.1 Technology stack

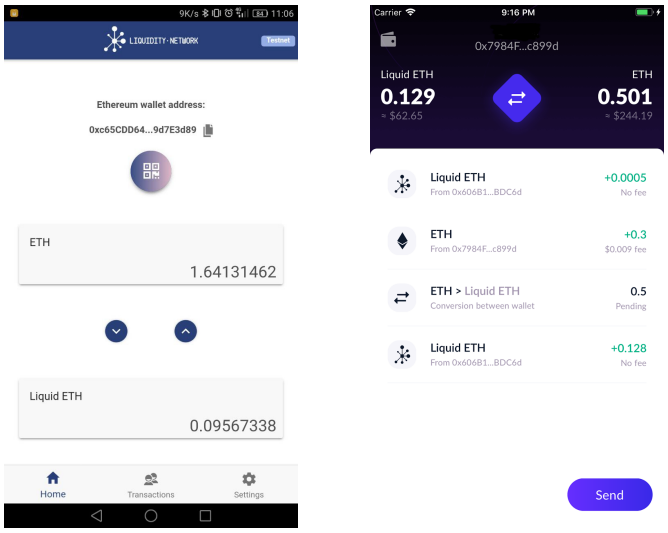
We wanted the application to be available for the two major mobile operating systems currently available, Android and IOS. However, because they have a very different architecture, they require to be developed in two different programming languages, Java and SWIFT or Objective-C. To avoid duplicating the work, a cross-platform framework was used where the application would be developed once and used in both OS. We consider two JavaScript frameworks Ionic and React Native, we will discuss here their pros and cons and explain the reasons that leaded the choice of React Native. Then we will discuss the chosen solution for State management.

5.0.1.1 Ionic

Ionic is a cross-platform framework maintained by the Apache Foundation released in 2013. It uses the TypeScript language which is a super-set of Javascript, it brings type safety to Javascript. The major difference with its competitor React native is that Ionic is a Hybrid framework. The user interface of a hybrid

[13]Liquidity Wallet playstore: `https://play.google.com/store/apps/details?id=com.liquiditynetwork.wallet`

[14]Appstore: `https://itunes.apple.com/ch/app/liquidity-network-wallet/id1395924630?l=en&mt=8`



(a) First version Ionic (b) Final version on React Native

Figure 11: Ionic and React native version of the app

framework runs in a Web View, the whole application is wrapped in a web browser engine that renders the interface. The use of native features of the device such as the Camera, Storage, etc ... require specific plugins to bridge native features to the Web View. Ionic is very easy to get started if the developer is familiar with web development. It is based on Angular, the most popular web development framework. With Ionic you can be fairly confident that the applications will render the same on both, Android and IOS as the UI components such as button, forms, scroll view are emulated within the Web View.
Ionic was originally chosen for the application, we were able to release a first fully functional version of the wallet within a month, without any prior knowledge in mobile development or JavaScript. However, the Hybrid application model quickly started to show its limitations. With the user interface getting more complex the performances of the Web View were affected, the reactivity was low and the latency quite high, especially on lower end devices. It was decided that this was not a sustainable solution for further developments and that a different approach was needed. Figure 11a is a screenshot of the first version of the app with Ionic.

5.0.1.2 React Native

React Native is an open source JavaScript framework maintained by Facebook. It is the mobile version of the famous web development framework React. React is meant for building user interfaces and advanced single-page applications for web browsers. React and React native for mobile devices were released in 2013 and 2015, respectively. Both use the JSX syntax to structure user interfaces.

These technologies rely on the virtual Document Object Model (virtual DOM), which is key to their performances. It keeps in memory a copy of the rendered screen, the virtual DOM. When the screen needs to be updated, the new DOM is compared to the virtual DOM and only the modified elements are updated on the screen. The only drawback of this approach is a higher memory consumption. As opposed to Hybrid apps like Ionic, React Native is a Native app. The application is not wrapped in a webview but it is rendered natively by the operating system. Components like Buttons and Forms are real Android or IOS components. The drawback is that the developer has to take care and adapt the application for each OS much more frequently than with Ionic. It even requires sometimes to write some native code (Java or Swift). However, the performances were much more satisfying. After a month developing on Ionic the choice of moving to react native was taken. Unfortunately most of the code had to be rewritten but the end result was more satisfying. Figure 11a is a screenshot of the home screen of the final version of the app.

5.0.1.3 State managment

React native does not provide any solution for the state management of the application. To fulfil this task the Redux library was slected. Redux implements a "single" store often call a "single source of truth". Redux attempts to make state mutations predictable by imposing certain restrictions on how and when updates can happen. The state is made read only and the only way to change the state is to emit an action. Reducers handle the action, they are just pure functions that take the previous state and return the next state and no side effect is allowed. Because all changes are centralised and happen one by one in a strict order, there are no race conditions to watch out for. Redux makes state management easier and more robust when there are many interactions happening in parallel.

5.0.1.4 Wallet features

In this section we will present the main features of the wallet

Deposit : It allows to make a deposit to the smart contract on the Ethereum main network of the NOCUST payment hub. A deposit consists of a contract call to a specific function with an amount of Ether.

Withdrawal : It allows to withdraw funds from the payment hub contract.

Off-chain transaction : Allow to make off-chain transaction without fees and near instant through the NOCSUT hub.

On-chain transaction : Regular on-chain transactions.

Secure private key storage : Private keys are stored locally encrypted in the device secure storage.

Private keys backup and recovery : Mechanism to backup keys offline allowing users to recover funds if their device is lost.

Hub monitoring : Verify the state of the hub server and the hub smart contract to insure that no cheating happened. Warn the user if any inconsistency is detected.

5.0.1.5 User interface

We will present the user interface and its most important screens. Note that the Fingerprint lock screen, PIN lock screen, Share wallet screen and Settings screens are not presented here. We will refer to figure 12.

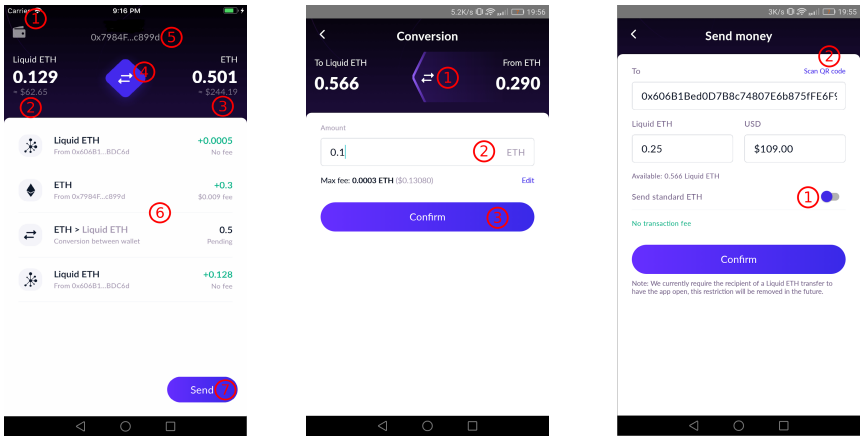
Home screen : On figure 12a we show the main screen of the application, it acts as a dashboard for the user. (1) Opens the share wallet screen, it allows the user the share its Ethereum address through a QR code or via copying it to the clipboard. (5) Is the short form of the wallet Ethereum address. (2) and (3) are the balances of the wallet, (3) is the on-chain balance of the account in the Ethereum Network. (2) Is the off-chain balance in the NOCUST payment hub. (4) opens the conversion screen that allows the user to transfer funds between the 2 balances. (6) is the transaction history of the wallet, it shows the on-chain and off-chain transactions and the conversions between the balances. (7) Allows you to access the send screen.

Conversion screen : This screen allows you to transfer funds between the 2 balances. From on-chain to off-chain, we call it a deposit and from off-chain to on-chain, we call it a withdrawal. (1) Is a button that allows you to change the direction of the conversion. In (2) you specify the amount to transfer and confirm with (3). Note that deposit and withdrawal are on-chain operations (contract calls) that cost gas fees.

Send screen : This screen allows you to make transactions by specifying the address and the amount in the forms. With (2) you can access the QR code scanner to scan an Ethereum address or a Payment request. By turning on the switch (1) you can send the transaction on-chain instead of the default off-chain. Note that by turning the switch on you will pay transaction fees.

5.1 Transaction flow

To make an off-chain transaction, the application currently requires the recipient and the sender to be online to sign their state update. This is a major drawback compared to regular on-chain transactions where no action is required from the receiver. However, in the future the system might be changed to support passive accepts requiring signature only for the state update of the sender.



(a) Home screen (b) Conversion screen (c) Send screen

Figure 12: Wallet main screens

The transaction is fast and shows up directly in the wallet of the receipient. However, it can sometimes need up to 20 seconds maximum to be settled. If the recipient of a transaction does not have the app open, it will receive a push notification and will have up to 2 minutes to accept the transaction.
Deposits and withdrawals are on-chain transactions, they require first to be accepted on the Blockchain. Once the recipient accepted a deposit, 60 blocks of confirmation are required before the deposit shows up in the wallet. A withdrawal takes much longer, given that to get the funds back on-chain you have to wait for a complete round or 2 commitments to pass. The reason is that you have to wait for the full transaction finality and therefore you need to go through the full challenge period. If the hub passes the challenge period successfully, you can initiate a second transaction to confirm your withdrawal and get your funds in the on-chain account.

5.1.1 Authenticated invoicing system

Most payments are expected to be made with online merchants. In order to make it convenient for users and merchants, a standard payment request format was designed. The payment request is supposed to fit in a QR code. The payment flow with an online merchant is the following:

1. The user checks out his cart on the online merchant website.

2. The online merchant shows the payment request in the form of a QR code.

3. The user scans the payment request with the mobile wallet.

4. The user verifies the address and the amount of the merchant and confirms the payment.

5. The merchant waits for completion of the transaction.

It is very convenient for users to use payment requests. The address and the correct amount is automatically filled in. The transaction is secured through the use of a secret random nonce passed in the payment request. This nonce is generated on the server of the merchant website and included in the QR code showed to the user. Then the secret nonce will be included in the signature of the user authorising the transaction and therefore passed to the hub. When the merchant receives the transaction, he should verify that the nonce matches. The nonce is used similarly to the authentication cookies in web browsers. This way the transaction is properly authenticated and it allows a binding between the merchant and the user sending the transaction.

5.2 Security

Security is a critical aspect for a cryptocurrency wallet. It holds the private keys allowing to unlock the funds and they need to be protected accordingly. In this section we will go over the different security features that the wallet offers.

5.2.1 Private keys protection

A first critical step is the key generation. Several cryptographic softwares in the past were found vulnerable due to an unsafe random number generator. If the initial random seed used to derive the private key is predictable, the users are exposed to a high risk of theft. To ensure safe random number generation the libraries chosen were carefully reviewed to ensure that they are suitable for highly critical cryptographic operation. For example, a common mistake in JavaScript is to use the convenient function `Math.random()`. This function should never be used for cryptographic operations, since it is only suitable for non security critical applications. We chose the react native plugin `react-native-randombytes`[23], which relies on the cryptographic module of the underlying operating system. For Android, the `SecureRandom`[2] function is used. In 2013 a flaw in its design was found[8] that lead to a theft of Bitcoins. However, the vulnerability was since then corrected. For IOS the function `SecRandomCopyBytes`[36] is used.
For protecting the private keys the wallet relies on the secure storage of their respective operating systems. Keys are encrypted at rest and require the device to be unlocked to use them. The keys used to encrypt private keys never reach the main application process. On recent devices that offer a Hardware Security Module, the keys are non exportable, making it impossible to retrieve the encryption keys even with root access to the device. The secure storage on Android is the KeyStore implementation[1]. It is fully available starting from Android

API level 22 (Released in Octobrer 2015). Therefore, on older devices the encryption keys are stored in the shared preference of the device system making rooted phones vulnerable. On IOS the implementation for secure storage is the Apple keychain[3].
To restrict access to the application and therefore protect the funds of the user, the application will be locked at start up and will require to be unlocked. When creating a wallet, the user is requested to choose an authentication method. Currently, two authentication methods are available: a 6 digits PIN code or the fingerprint sensor. The fingerprint sensor option will obviously be available only on devices that have one. On Iphone X the Fingerprint lock method will be automatically replaced by Face ID, the face recognition lock screen from Apple. Note that face ID and the fingerprint authentication have to be configured on the underlying OS, respectively IOS or Android to be used as a authentication method within the app. The lock screen will be showed every time at start up. If the application is inactive or put in the background for more than 5 minutes, it will be locked automatically. In the future, we plan to add a regular password lock screen as an additional lock method.

5.2.2 Wallet recovery

The wallet follows the Bitcoin standard BIP39[5], BIP32[4] and BIP44[6] for the key derivation. 128 bits of entropy from the random generator described in the previous section are used and a 4 bits checksum is added. This is then encoded using 12 words selected in a list of 2048 regular English words. The word list specified by the BIP39[5] standard is constructed to avoid confusion so similar words are avoided. The 12 words consitue the passphrase from which the private key is derived and it should be kept secret at any time. BIP32[4] defines the key derivation function that should be used and the hierarchical derivation scheme that supports several cryptocurrencies and allows to create several key pairs. BIB44[6] defines the derivation path to use depending on the cryptocurrency. For the wallet we used the BIP32 derivation path `m/44'/60'/0'/0/0`, 44 is a version identifier, 60 is the identifier for the Ethereum main network.
It is convenient to write down a passphrase on a sheet of paper as a permanent offline backup. Additionally, the passphrase includes a checksum to ensure that it was not mistyped. If the user looses his phone or it is stolen, the offline backup will allow to recover the funds. When setting up the wallet, two options are offered to the user: create a new wallet from a newly generated random seed or restore a wallet using an existing passphrase.

5.2.3 Blockchain source

To get blockchain data and broadcast transactions we currently rely on Infura infrastructures[13]. It is a free service run by the Ethereum foundation that offers open RPC endpoints of Ethereum nodes. It is a third party in which a certain amount of trust is required, the trust model is worse than for a SPV client. They could potentially setup double spending attacks by simply lying

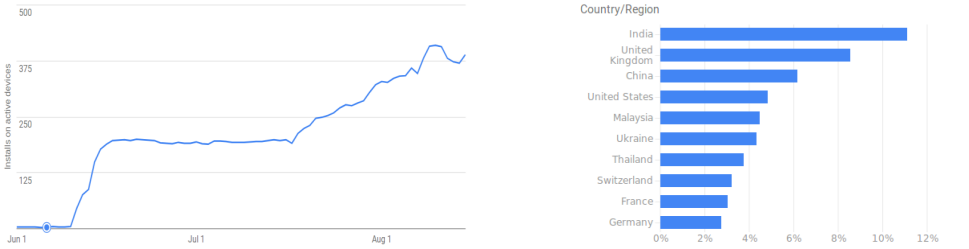
to the wallet. The data from the Ethereum network is not verified through the verification of membership proof in the state Merkle tree or transaction tree. In the future it is planned to allow the user to configure his own blockchain source. This would allow you to use the wallet with your own Ethereum node. For a better trust model, a SPV light client could be run directly on the phone. The go-ethereum library[11] can be used in light mode to sync only the block headers. This approach will be considered for future developments.

5.2.4 Hub monitoring

The security of NOCUST payment hubs rely on the the users carefully monitoring the contract state and if any inconsistencies are detected an on-chain challenge should be initiated. The wallet automatically take care of the monitoring, if the hub appear to cheat or to be malfunctioning the user will get an alert. The current version does not allow to start a challenge from the wallet, an external tool has to be used[16]. The current round length is 18 hours, to insure the integrity of his off-chain account the user should come online at least once every 18 hours to verify the last checkpoint and start a challenge if necessary. In future developments of the hub it will be possible to start challenges on the behalf of someone else account. It will allows for third party auditors that can initiate challenges them-self for any account. Given the assumption that there is at least one third-party auditor somewhere that correctly verify the contract state and that is willing to react if he founds inconsistencies the online presence requirement for users will be relaxed. The audit mechanism can act as an additional security for the users.
If a challenge was started and that the hub appear to not be able to answer the challenge the contract will enter in recovery mode. The recovery mode makes the last valid commitment final, no more commitment are allow and off-chain payments are therefore frozen. The last valid commitment is the last commitment for which they was no challenge that the hub was not able to answer. In recovery mode everyone can withdraw their funds by providing a valid proof of stake (merkle proof of membership to the last valid commitment). The funds from the off-chain transactions that were made after the last valid commitment didn't reached full finally. Therefore they are reverted unless the hub provided enough collateral to guarantee these transactions. A separated contract call is required to recover these funds.

5.3 Application usage

We focused on delivering the application for Android. Even if both versions Android and IOS are fully functional, the listing process for Android applications on the Playstore is much simpler than for IOS. In this section we present an analysis of our audience and adoption. Then the results of a short user study will be presented.



(a) Active installs

(b) Geographic location of app installs

Figure 13: Active installs and geographic location

5.3.1 Analytics

The application integrate the Firebase analytic module from Google[**?**] to track the usage of the app. With Firebase we can look at metrics to see who and how the app is used. Data like number of sessions or retention is available to the developer. Additionally, some more personal informations coming from users Google accounts is anonymise and made available for the developer. We can see location, age, gender or centers of interest. Additionally, we setup application specific trackers. Firebase provides the possibility to add custom events to count the number times that the users fulfil certain tasks, for example, counting the number of deposits. All the results presented here are as of the 15th of August 2018.

Since its release in early June 2018 We counted 1192 first opens of the applications, figure 13a shows the number of active installs. It's the number of devices on which the application is installed and has been used in the last 30 days. There is currently about 400 active installs, we therefore account for a large number of uninstalls or inactive applications. We counted a total of 4432 sessions or number of openings of the app. This means that on average each user opened the app 3.7 times. The most popular location of app installs is surprisingly India. This can be explained by the large population and the relatively high English literacy in this country. It is followed by the United Kingdom and China. 84% of the users are male and the most popular age group is the 25-34 years with 36% followed by the 18-24 and 35-44 years with both about 17%.

To allow the users to try the application rapidly without having to make a deposit we setup a faucet to allow the user to get 100 wei for free. One wei is equal to $10^{-}18$ Ether, it is an extremely small amount of money but it is sufficient to demonstrate the capabilities of the wallet. The user can send amounts as small as 1 wei to other users without having to pay fees. The same transaction on-chain would make very little sense as the transaction fees would be around 0.10 USD at current gas prices. We counted 545 outgoing transactions from the wallet, unfortunately, only 130 unique users were able to make a transaction which is about 10% of the installs. Table **??** is a summary of some events that are tracked in the application.

	Count	Unique user
Off-chain transaction	545	130
On-chain transaction	17	13
Deposit	138	51
Withdrawal	5	3

Table 3: Use of the wallet functionalities

Regarding the lock screen method, 65% chose the 6 digits PIN code and 35% the fingerprint sensor. 79%, chose to create a new wallet and 21% chose to restore an already existing wallet.

5.3.2 User study

A short user study was conduced to get the expectations of users and feedback for further improvements. Answers from 30 app users where gathered during a survey period of one week.
First, it was needed to know more about the audience. We asked about their background, if they where rather into Engineering, Finance, Law or others. A large majority of our users unsurprisingly belonged to the "Software, Engineering and Research" category with 59% of the answers. The second category was "Finance, Business , Banking" with 13% and the third was "Marketing, Public relation, HR, Administration" with 9% of the answers. It was then asked to evaluate how familiar they were with Blockchain technologies on a scale from 1 to 5. With 1 being complete beginner and 5 expert. The average score was of 3.5/5, surprisingly the respondents from the technical category considered themself less familiar with Blockchain technologies than the non-technical categories. The category "Software, Engineering and Research" got on average 3.3 and the non-technical categories got on average 3.7. These answers overall confirm our expectation that our audience is rather technical and familiar with Blockchain technologies. Therefore, most of them probably already used a cryptocurrency wallet.
An important answer that we were looking for is whether the wallet should support exclusively off-chain transactions or both on-chain and off-chain transactions. Once third-party auditors would be available to protect other users accounts it could be possible to operate a wallet and transact exclusively off-chain and never owning any funds on-chain. The wallet could delegate the broadcast of withdrawal and thus paying the gas fees in off-chain ether. Supporting exclusively off-chain Ether would simplify the UI and improve the user experience. The user would not be required to manage his funds between two balances. Such a system would almost completely abstract the underlying complexity of the NOCUST hub.
In the user study it was first asked if they found the wallet too complex, 80% answered no. It was then asked if they understood why the wallet has 2 balances, on-chain balance and off-chain balance, 83% answered "Yes". It seems that that

these user fairly understood this concept of double balance in wallet. The next step was trying to understand if they would have preferred to see only one balance. The same question was asked but rephrased in different manners to better understand their needs. It was asked if they find it annoying to have to manage 2 balances, here, the responses were more balanced with 53% No and 46% Yes. Then "Do you think the wallet should have only one balance ?" 16% No, 40% Yes and 33% didn't care. In these 2 last questions the answers seems to be shared equally between the one balance and the two balance solution. However, when we asked "Do you think that the liquidity wallet should support both, on-chain and off-chain payments ?" the response was clear, 76% sais yes, in favour of the 2 balances solution. Some answers are contradictory, they don't seems to be really happy of having 2 balances but the majority wants the wallet to support both on-chain and off-chain. A solution might be to offer 2 modes, advanced with manual funds management and normal mode exclusively for off-chain. It has to be kept in mind that these users are probably fairly comfortable with Blockchain technologies and understands the concepts of on-chain and off-chain, the responses might be different with less experienced users.
It was asked to grade the overall user experience of the wallet, the average is at 3.4 out of 5 with the median at 4. We then asked to grade how secure they feelt while using the wallet, the average was at 3.3 out of 5 with a median at 3. No consequent differences were observed between user with a technical background and those that don't.
Out of curiosity, it was asked to grade how much they value the importance of open source software for cryptocurrency wallets. The average here was at 3.8/5. However, the average for technical users was at 4.3 and the average for non-technical users was at 3.8. Therefore, we can say that technical users are more sensible to open source than non-technical.

5.4 Further work

Support for several private key formats : To allow compatibility with other wallets and allow to import private keys from them we would need to support the import of several private keys frormat. We currently only support 12 words passphrases, however some users might want to import raw ECDSA private keys or the JSON format[27] use by other major wallets such as My Ether Wallet[18].

Support for ERC20 tokens : ERC20 is a standard[10] to create independent tokens or new cryptocurrency on the top of the Ethereum platform. It allows for a seamlessly support of the token in the wallet. ERC20 consist of a smart contract interface that a smart contract need to implement to creat a ERC20 tokens. The interface has 9 functions, the most importants are `balanceOf(address owner)` that allows to query the current token balance of address `owner` and `transfer(address to, uint256 value)` that allow to initiate a transfer of amount token `value` to the destination address `to`. The same way that a NOCUST hub handle Ether accounts it

could handle any token build on Ethereum and could therefore be added to the wallet.

Decentralised exchanges : A NOCUST hub supporting several tokens could authorise atomic swaps between two users. Exchanging an amount of token `A` for an amount of token `B`. Such swap transaction would have the same trust model as a regular transaction in a NOCUST hub. Existing cryptocurrency exchanges require users the withdraw the funds they wish to exchange to the accounts of the exchange platform. They will hold users funds for a short period of time to execute the trade before returning the funds. A NOCUST exchange do not require at anytime to take custody of its users funds therefore having a better trust model. Such a swapping feature could be added to the wallet when it will support ERC20 tokens.

support for Bitcoin : Bitcoin limited scripting languages unfortunately does not offer sufficient capabilities to run an instance of the NOCUST smart contract on-chain. However solution such as XCLAIM[50] would allow to have bitcoin backed token on Ethereum, meaning an ERC-20[10] compliant token that

Continuous integration : Due to tight time constrains to develop the application the development of automated tests and setup of proper a continuous integration pipeline was skipped. Continuous Integration best practices are crucial for the longer term maintenance of the application. Automated unit tests need to be develop to insure the correct behaviour of each module of the application. Additionally End-to-End or E2E tests need to be developed. E2E tests simulate the interaction of user of application, it allow testing the application close to real usage conditions. Libraries such as Detox[9] allow for such tests scenarios. The test suite would then need to be integrated within a Continuous integration pipeline running the tests after each commit to the source repository.

6 Conclusion

We looked at the Lightning network topology. Since it was introduced on the Bitcoin main network in January 2018 the Lightning Network is growing very fast. Its original authors were at first predicting a distributed, peer to peer topology for the network, while the current reality is that such a structure is not practical and scalable. The current network is more centralised than previously expected, the top 10 nodes are involved in 36.6% of the total number of channels and represent 55.5% of the node capacity. The network is currently mostly used for testing, users open very small channels to not take big risks. We have seen that fees are insignificant at the moment and routing nodes does not mind being unprofitable. Therefore, we can conclude that currently users of the Lightning Network are not yet acting like rationally actors. The topology will evolve as the network will gain traction and real life usage. Payment failures are major

issues at the moment and payments fail because of unsuccessful routing or lack of liquidities, therefore highly impacting the user experience. Given the current low transaction volume on the lightning network and the small number of merchants accepting Lightning payments the collateral can easily be artificially provided to prevent channels from being exhausted. On the receiving end of a linked payment, collateral requirements prevents the user from receiving an amount of money before he effectively first spent this amount. To our knowledge, there is few research ongoing to improve the liquidities issues, especially how to address the limitation of inbound capacity of a wallet.
The lightning network presents other issues such as the online presence requirement and the complex channel management. However, these issues will be addressed in the future by developments like Watchtowers and Autopilot feature.
We have seen that there is currently no practical mobile wallet for the Lightning Network. The only exception is the Bitcoin Lighting Wallet that opted for a drastic trade-off in terms of trust assumptions.
Payment channel hubs such as NOCUST on Ethereum allow for a better user experience. These payment hubs leverage the power of a Turing complete virtual machine to allow for more flexibility. In the future it is expected to have several hubs connected by payment channels. Therefore, we assume from the start a more centralised topology similar to the one towards which the Lightning Network is evolving.
We developed from scratch the first mobile wallet for Ethereum with full off-chain capabilities. A special care was given to the security of the wallet and more specifically the protection of the private keys. The wallet is available for Android and IOS and currently has about 1600 active installs as of early September 2018. We believe that this wallet provides a better user experience than existing mobile wallets supporting off-chain payments.

References

[1] Android keystore. `https://developer.android.com/training/articles/keystore`. Accessed: 14-09-2018.

[2] Android random number generator. `https://developer.android.com/reference/java/security/SecureRandom`. Accessed: 14-09-2018.

[3] Apple keychain. `https://developer.apple.com/documentation/security/keychain\_services`. Accessed: 14-09-2018.

[4] Bip32. `https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki`. Accessed: 27-08-2018.

[5] Bip39. `https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki`. Accessed: 27-08-2018.

[6] Bip44. https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki. Accessed: 27-08-2018.

[7] Celer network. https://www.celer.network/. Accessed: 27-08-2018.

[8] Cve-2013-7372. https://nvd.nist.gov/vuln/detail/CVE-2013-7372. Accessed: 27-08-2018.

[9] Detox, end-to-end testing and automation framework. https://github.com/wix/detox. Accessed: 27-08-2018.

[10] Erc-20 token standard. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md. Accessed: 27-08-2018.

[11] go-ethereum client. https://github.com/ethereum/go-ethereum. Accessed: 27-08-2018.

[12] How many people use bitcoin. https://www.bitcoinmarketjournal.com/how-many-people-use-bitcoin/. Accessed: 27-08-2018.

[13] Infura, scalable blockchain infrastructure. https://infura.io/. Accessed: 14-09-2018.

[14] Interactive lightning network explorer. https://lnmainnet.gaben.win/. Accessed: 31-08-2018.

[15] Lightning network bolt specifications. https://github.com/lightningnetwork/lightning-rfc. Accessed: 27-08-2018.

[16] Liquidity web wallet. https://wallet.liquidity.network/. Accessed: 27-08-2018.

[17] Loom network. https://loomx.io/. Accessed: 31-08-2018.

[18] My ether wallet. https://www.myetherwallet.com/. Accessed: 27-08-2018.

[19] Neo - an open network for smart economy. https://neo.org/. Accessed: 27-08-2018.

[20] Olympus server repo. https://github.com/btcontract/olympus. Accessed: 27-08-2018.

[21] Omisego. https://omisego.network/. Accessed: 31-08-2018.

[22] Raiden network. https://raiden.network/. Accessed: 27-08-2018.

[23] react-native-randombytes plugin. https://github.com/mvayngrib/react-native-randombytes. Accessed: 14-09-2018.

[24] Satoshi nakamoto talking about payment channels. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002417.html. Accessed: 27-08-2018.

[25] Spankchain. https://spankchain.com/. Accessed: 27-08-2018.

[26] Trinity, universal off-chain scaling solution. https://trinity.tech/#/. Accessed: 27-08-2018.

[27] Utc json keystore file. https://theethereum.wiki/w/index.php/Accounts,_Addresses,_Public_And_Private_Keys,_And_Tokens#UTC_JSON_Keystore_File. Accessed: 27-08-2018.

[28] open source community Acinq. Eclair, ln implementation written in scala. https://github.com/ACINQ/eclair. Accessed: 31-08-2018.

[29] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[30] open source community Blockstream. C-lightning, ln implementation written in c. https://github.com/ElementsProject/lightning. Accessed: 31-08-2018.

[31] Rasmus Dahlberg, Tobias Pulls, and Roel Peeters. Efficient sparse merkle trees. In *Nordic Conference on Secure IT Systems*, pages 199–215. Springer, 2016.

[32] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[33] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment channels over cryptographic currencies. Technical report, IACR Cryptology ePrint Archive, 2017: 635, 2017.

[34] William Entriken. A standard interface for non-fungible tokens. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md. Accessed: 31-08-2018.

[35] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 3–16. ACM, 2016.

[36] Apple Inc. Ios random number generator.

[37] VISA Inc. Operational performance data. https://s1.q4cdn.com/050606653/files/doc_financials/2018/Q3/Visa-Inc.-2018-Operational-Performance-Data.pdf, 2018. Accessed: 27-08-2018.

[38] Rami Khalil and Arthur Gervais. Revive: Rebalancing off-blockchain payment networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 439–453. ACM, 2017.

[39] Rami Khalil and Arthur Gervais. Nocust - a non-custodial 2nd-layer financial intermediary. Cryptology ePrint Archive, Report 2018/642, 2018. https://eprint.iacr.org/2018/642.

[40] Lightning Labs. Neutrino client. https://github.com/lightninglabs/neutrino. Accessed: 27-08-2018.

[41] open source community Lightning Labs. Lightning network daemon, ln implementation written in go. https://github.com/lightningnetwork/lnd. Accessed: 31-08-2018.

[42] Patrick McCorry, Surya Bakshi, Iddo Bentov, Sarah Meiklejohn, and Andrew Miller. Pisa: Arbitration outsourcing for state channels.

[43] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. Sprites: Payment channels that go faster than lightning. *arXiv preprint arXiv:1702.05812*, 2017.

[44] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[45] M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. *IEEE Journal on Selected Areas in Communications*, 23(1):89–103, Jan 2005.

[46] Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, 2017.

[47] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *draft version 0.5*, 9:14, 2016.

[48] open source community Satoshi Nakamoto. Bitcoin core client, original bitcoin client written in c++. https://github.com/bitcoin/bitcoin/. Accessed: 31-08-2018.

[49] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccd - 2017-08-07). https://ethereum.github.io/yellowpaper/paper.pdf, 2017. Accessed: 27-08-2018.

[50] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. Xclaim: Interoperability with cryptocurrency-backed tokens. Cryptology ePrint Archive, Report 2018/643, 2018. https://eprint.iacr.org/2018/643.